UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Dipartimento di Scienze Fisiche,
Informatiche e Matematiche

Assignment

LICM

A partire dal codice della precedente esercitazione implementare un passo di **Loop-Invariant Code Motion (LICM)**

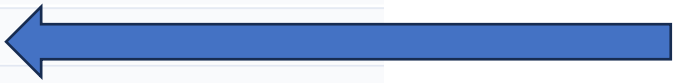
- Non usare l'acronimo LICM per il passo
  - Andrebbe in conflitto col passo ufficiale LLVM

LLVM Pass Manager

- Specificamente per i LOOP_PASS possiamo sfruttare la classe `LoopStandardAnalysisResults`:

Public Attributes

AAResults &	AA
AssumptionCache &	AC
DominatorTree &	DT
LoopInfo &	LI
ScalarEvolution &	SE
TargetLibraryInfo &	TLI
TargetTransformInfo &	TTI
BlockFrequencyInfo *	BFI
BranchProbabilityInfo *	BPI
MemorySSA *	MSSA



LLVM Pass Manager

- Specificamente per i LOOP_PASS possiamo sfruttare la classe `LoopStandardAnalysisResults`:

```
#include "llvm/IR/Dominators.h

PreservedAnalyses
LoopWalk::run(Loop &L, LoopAnalysisManager &AM,
              LoopStandardAnalysisResults &AR,
              LPMUpdater &U) {

  …
  DominatorTree &DT = AR.DT;
  BasicBlock *BB = (DT.getRootNode())->getBlock();
  …

}
```

Code motion

- Una volta identificate le istruzioni candidate per la code motion posso spostarle nel blocco preheader
- Vari metodi della classe `Instruction`:

- **`void   removeFromParent ()`**
- **`InstListType::iterator   eraseFromParent ()`**
- **`void   insertBefore (Instruction *InsertPos)`**
- **`void   insertAfter (Instruction *InsertPos)`**
- **`void   moveBefore (Instruction *MovePos)`**
- **`void   moveAfter (Instruction *MovePos)`**
- …

Code motion

- Si può selezionare, ad esempio, la prima o l'ultima istruzione di un blocco.
- metodi della classe `BasicBlock`:

Es.

```
Instruction *  getTerminator ()
```